PATENT ABSTRACTS OF JAPAN

(11)Publication number : 07-219789
(43)Date of publication of application : 18.08.1995

(51)Int.Cl. G06F 9/46

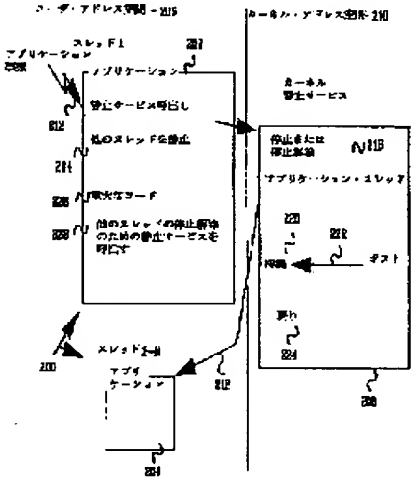
(21)Application number : 06-262140
(22)Date of filing : 26.10.1994

(71)Applicant : INTERNATL BUSINESS MACH CORP <IBM>
(72)Inventor : AULT DONALD F
ERNEST SCOTT BENDER
JOHN KEVIN FRANKS
JOHN ARTHUR HELMBOLD

(30)Priority
Priority number : 94 187675 Priority date : 27.01.1994 Priority country : US

(54) METHOD FOR PROCESSING EXTERNAL EVENT IN PLURAL THREAD SYSTEMS

(57)Abstract:
PURPOSE: To provide a method for processing an external event through a 1st thread by adjusting the stop of additional thread of plural thread applications.
CONSTITUTION: In response to the detection of the external event, the 1st sled transmits the event of stop for stopping that execution to the other additional thread of the application. Each thread does not hold severe resources. Therefore, when it is judged that it is safe to stop, each thread stops. The 1st sled is restarted before the stop of the final thread, and it is possible to perform an important operation freely without being interferred from the other additional threads. When the stop is an interruptive type, the interrupted execution of the other additional thread is restarted as soon as the important operation is completed, and the application recovers its ordinary operation.



LEGAL STATUS

[Date of request for examination]	26.10.1994
[Date of sending the examiner's decision of rejection]	
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]	
[Date of final disposal for application]	
[Patent number]	2856681
[Date of registration]	27.11.1998
[Number of appeal against examiner's decision of rejection]	
[Date of requesting appeal against examiner's decision of rejection]	
[Date of extinction of right]	

(19)日本国特許庁（ＪＰ）　(12) **公開特許公報（Ａ）**　(11)特許出願公開番号

特開平7－219789

(43)公開日　平成７年(1995)８月18日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ　9/46	３４０　Ｂ	7629－5B		

審査請求　有　　請求項の数14　ＯＬ　（全 12 頁）

(21)出願番号　特願平6－262140

(22)出願日　平成６年(1994)10月26日

(31)優先権主張番号　１８７６７５

(32)優先日　1994年１月27日

(33)優先権主張国　米国（ＵＳ）

(71)出願人　390009531
インターナショナル・ビジネス・マシーンズ・コーポレイション
ＩＮＴＥＲＮＡＴＩＯＮＡＬ　ＢＵＳＩＮＥＳＳ　ＭＡＳＣＨＩＮＥＳ　ＣＯＲＰＯＲＡＴＩＯＮ
アメリカ合衆国10504、ニューヨーク州　アーモンク　（番地なし）

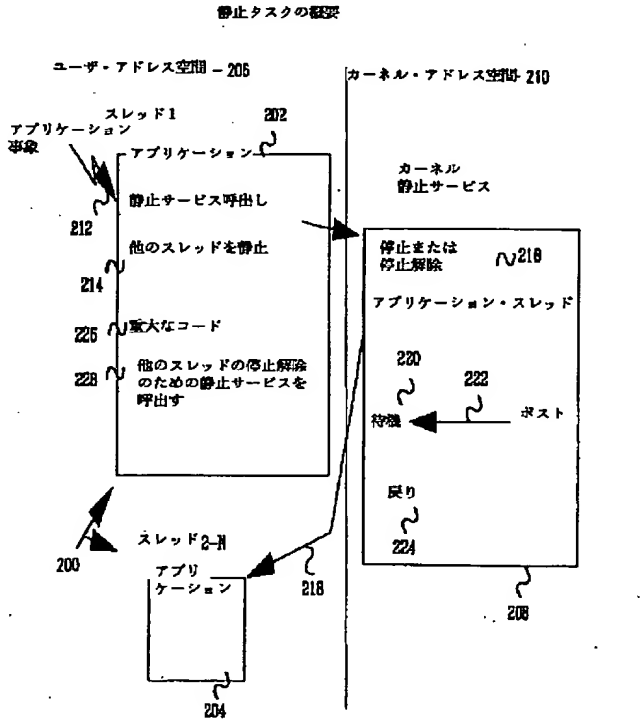
(72)発明者　ドナルド・フレッド・オールト
アメリカ合衆国12538ニューヨーク州ハイド・パーク、ルーズベルト・ロード　115

(74)代理人　弁理士　合田　潔　（外２名）

最終頁に続く

(54)【発明の名称】　複数スレッド・システムにおける外部事象を処理する方法

(57)【要約】
【目的】複数スレッド・アプリケーションの付加スレッドの静止を調整して、第１のスレッドによる外部事象の処理を行う方法を開示する。
【構成】外部事象の検出に応答して、第１のスレッドは、アプリケーションのその他の付加スレッドへ、その実行を中断するため静止事象を送信する。各スレッドは重大な資源を保持せず、従って静止することが安全であると判断される場合に、静止する。最後のスレッドが静止する前に第１のスレッドは再開され、他の付加スレッドからの干渉なしに重大な動作を自由に行うことができる。静止が中断タイプであれば、重大な動作の完了次第、中断した他の付加スレッドの実行を再開し、アプリケーションはその正常動作に戻る。



【特許請求の範囲】
【請求項１】第１のスレッドと第２のスレッドが共通アドレス空間で並列的に実行されるコンピュータ・システムにおいて、前記第１のスレッドによって外部事象を処理するための方法であって、
前記第２のスレッドを静止させるため、前記第１のスレッドから前記第２のスレッドへ静止事象を送信するステップと、
前記第２のスレッドへ送信された静止事象に応答して前記第２のスレッドが動作を静止するまで、前記第１のスレッドの実行を中断するステップと、
前記第２のスレッドへ送信された静止事象に応答して前記第２のスレッドが動作を静止した時、前記事象を処理するため前記第１のスレッドの実行を再開するステップと、
を含む方法。
【請求項２】前記第２のスレッドは、前記アドレス空間において前記第１のスレッドと並列に実行される複数の付加スレッドの１つであり、
前記第１のスレッドから前記複数の付加スレッドの各々に前記静止事象が送信される、
請求項１記載の方法。
【請求項３】前記複数の付加スレッドが当該スレッドへ送信された静止事象に応答して動作を静止するまで、前記第１のスレッドの実行が中断される、
請求項２記載の方法。
【請求項４】静止すべき前記複数の付加スレッドの最後のスレッドが、前記第１のスレッドの実行を再開する、
請求項３記載の方法。
【請求項５】前記第１のスレッドによる前記外部事象の検出に応答して、前記静止事象が前記第１のスレッドから前記第２のスレッドへ送信される、
請求項１記載の方法。
【請求項６】前記静止事象が、前記第２のスレッドを終了させるための終了事象である、
請求項１記載の方法。
【請求項７】前記静止事象が、前記第２のスレッドを中断させるための中断事象である、
請求項１記載の方法。
【請求項８】静止事象を前記第１のスレッドから前記第２のスレッドへ送信する前記ステップが、前記第２のスレッドの実行に割り込み、静止出口ルーチンに制御を与えるステップを含む、
請求項１記載の方法。
【請求項９】前記静止出口ルーチンが、前記第２のスレッドがシステム環境に影響を及ぼす程重大な資源を保持しているか否かを判断し、前記第２のスレッドが重大な資源を保持していないと判断する場合、前記第２のスレッドを静止させる、
請求項８記載の方法。
【請求項１０】前記第２のスレッドがシステム環境に影響を及ぼす程重大な資源を保持しているか否かを判断するステップと、
前記第２のスレッドが重大な資源を保持していないと判断する場合、前記第２のスレッドを静止させるステップと、
を更に含む請求項１記載の方法。
【請求項１１】前記第２のスレッドを静止させる前に前記第２のスレッドによって保持される重大な資源を解放するステップを更に含む、
請求項１０記載の方法。
【請求項１２】第１のスレッドと第２のスレッドが共通アドレス空間で並列的に実行されるコンピュータ・システムにおいて、前記第１のスレッドによって外部事象を処理するための方法であって、
前記第２のスレッドを中断させるため、前記第１のスレッドから前記第２のスレッドへ中断事象を送信するステップと、
前記第２のスレッドへ送信された中断事象に応答して前記第２のスレッドが動作を中断するまで、前記第１のスレッドの実行を中断するステップと、
前記第２のスレッドへ送信された中断事象に応答して前記第２のスレッドが動作を中断した時、前記事象を処理するため前記第１のスレッドの実行を再開するステップと、
前記第１のスレッドによる前記事象の処理の後、前記第２のスレッドの実行を再開するステップと、
を含む方法。
【請求項１３】複数の付加スレッドが前記アドレス空間において前記第１のスレッドと並列に実行され、
前記第１のスレッドから前記複数の付加スレッドの各々に前記中断事象が送信される、
請求項１２記載の方法。
【請求項１４】前記複数の付加スレッドが当該スレッドへ送信された中断事象に応答して動作を中断するまで、前記第１のスレッドの実行が中断される、
請求項１３記載の方法。

【発明の詳細な説明】
【０００１】
【産業上の利用分野】本発明は、複数スレッドを実行するプロセスにおける種々のスレッドの静止（すなわち、終了または中断）を調整する方法に関するものである。
【０００２】
【従来の技術】コンピュータのオペレーティング・システム、すなわち、ユーザ・アプリケーションとハードウェアの間のインターフェースをとり、コンピュータ・システムの基本的監視機能を遂行するソフトウエアは当業者によく知られている。多くの現在のオペレーティング・システムは、１つのプロセスまたはアプリケーションの中で複数のスレッドを許容する。複数スレッド・アプ

リケーションとは、その仕事を実行するため１つ以上のスレッドの制御を使用するプログラムと定義される。（本明細書において、用語「プロセス」と「アプリケーション」は、同義語として使われ、両者とも、共通アドレス空間を共有する１つ以上のスレッドを参照する。）A. S. Tanenbaum著Modern Operating Systems, (1992)において、いくつかの現在のオペレーティング・システムが一般的に説明され、特にその５０７頁から５２３頁においてスレッドが記述されている。

【０００３】複数スレッド・アプリケーションをサポートするオペレーティング・システムの特定の例は、最近発表されたＭＶＳ拡張機能オープン・エディション（ＯｐｅｎＥｄｉｔｉｏｎ）を持つＩＢＭ ＭＶＳ／ＥＳＡオペレーティング・システムである。ＭＶＳ拡張機能ＯｐｅｎＥｄｉｔｉｏｎを用いることによって、ＩＢＭシステム／３９０コンピュータとＭＶＳ／ＥＳＡオペレーティング・システムからなるハードウェア／ソフトウェア・プラットホーム上で動作し、かつ、ＩＥＥＥ ＰＯＳＩＸ １００３．１、１００３．２および１００３．４ドラフト標準に準拠したアプリケーションをプログラムすることができる。（ＩＢＭ、ＯｐｅｎＥｄｉｔｉｏｎ、ＭＶＳ／ＥＳＡおよびＳｙｓｔｅｍ／３９０は、インターナショナル・ビジネス・マシーンズ・コーポレイションの登録商標である。）ＭＶＳ拡張機能ＯｐｅｎＥｄｉｔｉｏｎの更に多くの情報は、本明細書で引用しているが、下記の刊行物で参照することができる：Ault著 "Fork Clone Address Space Implementation on MVS", IBM Technical Disclosure Bulletin, vol. 35, no. 6, pp. 363-67 (Nov. 1992);Ault著 "Interoperability Between MVS and POSIX Functions", IBM TechnicalDisclosure Bulletin, vol. 35, no. 6, pp. 383-88 (Nov. 1992);Ault他著 "Cross-Address Space Control Function", IBM Technical Disclosure Bulletin, vol. 36, no. 10, pp. 591-95 (Oct. 1993);Introducing OpenEdition MVS, ＩＢＭIBM営業資料番号 GC233010-00(1993); MVS/ESA Support for IEEE POSIX Standards: Technical Presentation Guide, ＩＢＭ営業資料番号GG24-3867-0 (1993).上述のように、ＭＶＳ／ＥＳＡオペレーティング・システムのＭＶＳ拡張機能ＯｐｅｎＥｄｉｔｉｏｎは、１つのプロセス内で複数のスレッドの使用を許容する。ＭＶＳの用語において、スレッドは、１つのタスクとみなすことができる。従って、複数スレッドは、１つのＭＶＳアドレス空間内で複数のＭＶＳタスクを使用することに等しい。

【０００４】複数スレッド・アプリケーションは多くの状況において利点があるが、複数タスク（すなわち、複数スレッド）アドレス空間における適切なタスク制御が欠如しているため、終了、デバッグ作業およびダンプ処理の上で問題が起きる。このため、ＰＯＳＩＸ規格では、複数スレッドの１つが異常終了する場合当該プロセス内のすべてのスレッドの終了が求められる。これは、ＭＶＳでは、ジョブ・ステップ・タスクの停止または該当タスクを停止するＣａｌｌＲＴＭを使用することによって実行される。しかし、ＯｐｅｎＥｄｉｔｉｏｎのＭＶＳスレッドをサポートするＭＶＳタスクに上記のようなタイプの非同期停止を送信する場合、多くの問題に遭遇する。

【０００５】発生する１つの問題は、プロセスのスレッドが突然ランダムに停止される場合、実行時ライブラリが共通のプロセス資源のクリーンアップを逐次処理することができないというものである。別の問題は、２つの命令の間で停止されつつある構成要素の多くは十分なエラー回復機能を持っていないというものである。場合によっては、これらの欠陥は、例えばファイルシステムの破壊といった破局的結果を招くことがある。停止誤り回復手順は改善できるが、この種の停止全体を回避することが望まれる。

【０００６】制御された方法で複数スレッド・アプリケーションの残存スレッドを中断する必要性は、デバッグ作業において発生することがある。複数スレッド・アプリケーションをデバッグする際、デバッグを行うユーザは、実行するスレッドとある特定事象に対して中断すべきスレッドを選択でき、また、実行／中断状態を動的に変更することができることを必要とする。この中断プロセスは、また、アプリケーションの流れを変えるような種類のもの、あるいは、中断の時点で当該スレッドが重大なシステム管理資源を保持していることを許容するような種類のものであってはならない。

【０００７】複数スレッド・アプリケーションの残存スレッドを中断する必要性が発生する別の場合は、すべてのスレッドから得られる情報を用いて当該プロセスのダンプを行う時である。必要性は、上述したデバッグの状況と同様である。ダンプを要求するタスクは、呼び出し元タスクがダンプをとることを妨げるような重大なシステム資源をその他のアスクが保持していない場合そのようなその他のタスクのすべての実行を中断できなければならない。ダンプがとられたあと、ダンプを行うタスクは、他のタスクの実行を再開しなければならない。

【０００８】

【発明が解決しようとする課題】このように、複数タスク・アドレス空間において適切なタスク制御が欠如する場合は、終了、デバッッグおよびダンプの各局面において問題が発生する。複数スレッドが動作する環境において、予測可能でかつ非破壊的な形態でタスクの実行を終了あるいは中断させる仕組みが求められる。

【０００９】

【課題を解決するための手段】上述の課題は、あるスレッドが起動された時点で当該アドレス空間に存在するすべての他のスレッド（すなわちタスク）に静止事象（ＱＵＩＥＳＣＥ ＥＶＥＮＴ）を送信する新しい静止機能

を作成することによって解決される。静止機能を起動したスレッドは、すべての事象が実行され、目標スレッドが期待した状態に置かれるまで待機する。

【００１０】これを達成するため、静止事象を受信次第オペレーティング・システムがその制御権を渡すべき静止機能出口(ＥＸＩＴ)ルーチンを、ユーザがオペレーティング・システムに通知するための登録機能が提供される。出口が指定されていない場合は、該事象をいつ処理するかはオペレーティング・システムが決定する。

【００１１】事象の伝達は、目標スレッドの実行に割り込みを行うサービス要求ブロック／割込み応答ブロック（ＳＲＢ／ＩＲＢ）というブロックの組み合わせによる。割り込みの時点で動作していた要求ブロック（ＲＢ）に関し静止事象を実行させることが安全であるかどうかを検査する種々のチェックがなされる。これらのチェックには、制御を静止機能出口に渡すことがシステム環境にとって受容可能であるかを確認することが含まれる。例えば、システム・サービスに割り込むことは望しくないからである。制御が静止機能出口に渡され、静止事象が実行されるべきであると判断されると、適切な処置がとられる。もしも静止事象が終了のためのものであったとすれば、出口ルーチンは当該スレッドの実行を終了する。静止事象がスレッドを中断するべきものであったならば、出口は、適切な中断サービスを提供する。

【００１２】システムＩＲＢまたはユーザの静止出口が静止事象を実行できないと判断すると、事象は未処理のままにされ、次のシステム・サービスの出口に再び伝達されるか、あるいは、それより早くユーザが静止事象を実行できる安全点に達したことを検知すればその時点でシステム・サービスの出口に再伝達される。

【００１３】静止事象の伝達は、最後のスレッドが期待した状態に入るまですべてのスレッドに対して実施される。そのような最後のスレッドは、静止機能を起動した元のスレッドにポスト信号を送信し、該スレッドの待機を解く（すなわち，元のスレッドを再開する）。

【００１４】この解決方法の利点は、静止状態がいつスレッドに効果を及ぼす状態に入るべきかを決定することが可能であるという点である。重大な資源を保持しているスレッドを無条件に停止させるという問題は回避される。コードの「不安定」または「重大な」部分で実行中のスレッドを非同期的に停止させることに起因する破壊的な結果もまた回避される。

【００１５】

【実施例】図１は、本発明を組み入れているコンピュータ・システム１００の概要であり、静止機能の実現のための各システム層の間の関係を示している。これらの層は、図１の上部から始まって、アプリケーション層１０２、言語サブルーチン／実行時ライブラリー（ＲＴＬ）層１０４、オペレーティング・システムまたはカーネル層１０６、およびハードウェア層１０８である。

【００１６】システム１００の基部には、１つ以上の中央処理装置（ＣＰＵ）、主記憶装置、および、磁気ディスク記憶装置、磁気テープ装置、端末、プリンタ等の入出力装置から構成されるハードウェア層１０８がある。これらの構成要素は当業者にとっては周知のものであるので、特に図示しない。

【００１７】ハードウェア層１０８の上の層のカーネル層１０６は、ＣＰＵ、記憶装置および入出力装置を管理する基本ハードウェア層を制御し、ハードウェア層における資源へのアプリケーション・コードのアクセスを可能にする呼び出し可能サービスを含むソフトウェアから構成される。（本発明に関連した静止サービスは上記呼び出し可能サービスの一部である。）カーネル層１０６は、ＩＢＭシステム／３９０コンピュータからなるハードウェア上で動く拡張機能ＯｐｅｎＥｄｉｔｉｏｎを持つＭＶＳ／ＥＳＡオペレーティング・システムから構成される。しかし、本発明は、上記のようなハードウェア／ソフトウェア・プラットホームに限定されるものではなく、他のプラットホームを代替的に使うことができる。

【００１８】言語サブルーチン／実行時ライブラリ（ＲＴＬ）層１０４は、カーネル層１０６のすぐ上部に位置する。層１０４は、多くのアプリケーションによって使われる高級言語（ＨＬＬ）をサポートするルーチンから構成される。層１０４は、基本的には、要求されたＨＬＬ機能を、その要求を実行するために適切なカーネル・サービスに変換する。

【００１９】アプリケーション層１０２は、システム１００の最上部の層であって、１つ以上のユーザ・アプリケーションを含む。アプリケーションは、カーネル１０６から直接に、あるいは、言語サブルーチン／実行時ライブラリ１０４を経由してＨＬＬから、システム・サービスを要求することができる。本明細書の仕様の目的から、アプリケーション層１０２と言語サブルーチン／実行時ライブラリ１０４とは集合的に一括してアプリケーションとみなすことができる。

【００２０】図２は、システム１００において実施される本発明の静止機能の概略流れ図である。（停止、停止解除または終了を実行するする）特定の静止機能に関するステップはこの流れ図にすべて含まれてはいないが、すべての静止機能はこの一般的流れ図に従う。

【００２１】図２は、図１のシステム１００のアプリケーション層１０２のアプリケーション２００を示している。該アプリケーション２００は、第１のスレッド２０２（スレッド１）と１つ以上のその他の付加スレッド２０４（スレッド２からN)を含む。スレッド２０２、２０４は、多くの独立プロセス属性を持つ。例えば、それ自身のプログラム・カウンタと状態（すなわち実行中、待機または実行阻止中等の状態）をそれぞれのスレッドが持つが、 それらスレッドが共通のユーザ・アドレス

空間を共有するという点で独立プロセスとは相違する。（アプリケーション２００は、実際、同一アドレス空間を共有するスレッド２０２、２０４のセットとして定義することができる。）スレッド２０２は、必ずしも作成されるべきアプリケーション２００の最初のスレッドではない；それが静止サービスを定義して起動するが故に、他のスレッド２０４と単に区別するために第１のスレッドと呼ぶにすぎない。

【００２２】図２にはまた、システム１００のカーネル層１０６に配置された本発明の静止サービス２０８が示されている。静止サービス２０８は、ユーザ・アドレス空間とは別のカーネル・アドレス空間２１０に位置し、以下のような独自のサービス（または機能）を提供する:

１．quiesce_exit_registration（出口登録）サービス３０６：スレッドが以下にリストされる静止機能の１つを要求した時に制御を受け取るべき静止出口ルーチンのアドレスを初期スレッドが提供することを可能にする。静止出口は、登録した初期スレッドと関係づけられるスレッドで発生する静止事象の結果として制御を受け取るアプリケーション・レベル・ルーチンである。この出口は、伝達された静止事象を実行させる責任がある。

２．quiesce_freeze（停止）サービス４０４：すべての他のスレッドに中断事象を送信し、ユーザ定義静止出口ルーチンに制御を渡す。この事象に関し、静止出口ルーチンは、重大な資源がスレッド２０４によって保持されているか判定する。重大な資源が保持されていなければ、静止出口ルーチンは、quiesce-freeze-selfサービスを呼び出す。重大な資源が保持されている場合、重大な資源が開放されるまで、静止事象の処理は遅延される。

３．quiesce_freeze_self（自己停止）サービス７２２：起動元スレッドを中断する。このオプションは、中断事象の受信時に静止出口ルーチンによって使われる。

４．quiesce_event_put_back（処理遅延）サービス７３４：静止事象の処理を後の時点まで遅延させる。このオプションは、現在の実行環境の事情のために処理できない中断事象の受信時に静止出口ルーチンによって使われる。

５．quiesce_unfreeze（停止解除）サービス８０６：停止されたすべてのスレッドを再開する。

６．quiesce_term（終了）サービス９０４：すべての他のスレッドに終了事象を送信し、ユーザ定義静止出口がquiesce_exit_registrationサービスを用いて指定されている場合ユーザ定義静止出口に制御を渡す。この事象に関し、静止出口は、重大な資源がスレッド２０４によって保持されているか判断する。重大な資源が保持されていなければ、静止出口は、スレッドを終了させるためpthread-exitサービスを呼び出す。重大な資源が保持されている場合、重大な資源が開放されるまで、静止事象の処理は遅延される。

７．quiesce_force（バイパス）サービス：すべての他のスレッドへ終了事象を送信し、ユーザ定義静止出口への呼び出しをバイパスさせる。

【００２３】アプリケーション事象２１２がスレッド２０２上で検知される時、本発明の動作が始まる。アプリケーション２００の他のスレッド２０４が実行中は、このアプリケーション事象２１２は処理されることができない。事象２１２は、実施例によっては、停止、プログラム・チェックまたはブレーク・ポイントであり得る。この時点で、アプリケーション２００は、ダンプをとることができるように、あるいは、診断機能を実行することができるように、他のスレッド２０４の実行を中断することを必要とする場合がある。あるいはまた、アプリケーション２００は、他のスレッド２０４が終了しなければならないことを当該他のスレッドに通知することを必要とする場合もある。

【００２４】上述したように、デバッグ作業にも本発明を使うことができる。しかし、本発明はそのような使用に限定されることはなく、また、そのような使用の詳細は本発明の対象ではない。

【００２５】本発明に従って、スレッド２０２が、アプリケーション２００において他のスレッド２０４の静止を必要とする事象２１２を通知する時、アプリケーションの他のスレッドを静止するため、スレッド２０２は、カーネル静止サービス２０８を呼び出す（ステップ２１４）。静止サービス２０８は、静止通知２１８を他のスレッド２０４へ送信し（ステップ２１６）、静止通知が他のスレッドによって実行されるまで待機（すなわち中断）する（ステップ２２０）。静止通知２１８を受信次第、スレッド２０４は静止のタイプに基づいて該当する措置をとる。最後のアプリケーション・スレッド２０４は、静止通知を実行する時、カーネル静止サービス２０８で待機中のスレッド２０２にポスト信号を送る（すなわち、スレッド２０２を再開させる）。スレッド２０２へのポスト信号の送達とともに、静止サービス２０８は、アプリケーション２００のスレッド２０２へ制御を戻す（ステップ２２４）。

【００２６】スレッド２０２は、今や、他のスレッド２０４が動作中には処理することができなかった重大なコードを実行することができる（ステップ２２６）。この重大なコードの実行が完了し、当初の措置が他のスレッド２０４を停止させるべきものであったなら、スレッド２０２は、他のスレッドを停止解除するため静止サービス２０８を呼び出す（ステップ２２８）。

【００２７】図３は、アプリケーション２００が、静止事象を扱うアプリケーション定義ルーチン（図７から図９に記載）である静止出口をどのように登録するかを示す。以下詳細に説明する。

【００２８】静止出口を登録するため、アプリケーショ

ン２００の第１のスレッド３０２が、静止サービス２０８のquiesce_exit_registrationサービス３０６を呼び出し（ステップ３０４）、静止出口ルーチンのアドレス（quiesce_exit）を渡す。Quiesce-exit-registrationサービス３０６は、スレッド３０２を起動するためスレッド制御待ち行列エレメント（ＴＣＱＥ）３０８にアプリケーション静止出口アドレスを記憶する。図６に示されるように、ＴＣＱＥ３０８は、スレッド３０２とアプリケーション静止出口アドレス６０４と待機／ポストのための事象制御ブロック６０６とを識別するスレッド識別子６０２を含むメモリ上の定義された領域である。今や、スレッド３０２は静止事象に関して登録された。

【００２９】スレッド３０２は、カーネル・アドレス空間におけるスレッド作成（pthread_create）サービス３１４を用いて、その他の付加スレッド３１２を作成することができる（ステップ３１０）。カーネル・スレッド作成サービス３１４は、適切な方法で実施されるが、その方法は当業者には周知であるので、本明細書には記載しない。作成する新しいスレッド３１２（ステップ３１６）の各々に関して、スレッド作成サービスは、スレッド３０２に関するＴＣＱＥ３０８と同様のＴＣＱＥ３０８を構築し（ステップ３１８）、固有のスレッド識別子６０２を割り当て、作成元スレッド３０２のＴＣＱＥ３０８から新たに作成されるスレッド３１２のＴＣＱＥ３０８へ静止出口アドレス６０４をコピーする。

【００３０】かくして、アプリケーション中の全てのスレッド３０２、３１２が同一の静止出口アドレス６０４を持って登録された。図３で示されるように、各ＴＣＱＥ３０８は、適切な方法で（例えば、ポインタ、連続的メモリ・ロケーションまたはその他の方法で）次のスレッド３１２のためのＴＣＱＥ３０８に連結し、以下に記述のようにチェインまたはスレッド制御待ち行列（ＴＣＱ）３２０を形成し、このため、ある特定のアプリケーションに関するＴＥＱＥ３０８は逐次走査できる。

【００３１】図４は、図２の概略流れ図をより詳細にした本発明のプロセス流れ図である。始めに、アプリケーション２００中のすべての他のスレッド２０４の停止を必要とする事象２１２が、アプリケーション・スレッド２０２上で発生する。事象２１２を検知する時、アプリケーション・スレッド２０２は、静止サービス２０８のquiesce_freezeサービス４０４を呼び出す（ステップ４０２）。quiesce_freezeサービス４０４は次に内部事象生成ルーチン４０８を呼び出す（ステップ４０６）。

【００３２】アプリケーション２００中の起動元スレッド２０２以外のスレッド２０４の各々に関して、事象生成ルーチン４０８は、静止出口インターフェース・ブロック（ＱＥＩＢ）４１２を先ず作成する。これは、ＴＣＱ３２０をサーチして、すべての今後のスレッド２０４を識別することによって達成される。図５を参照すると、ＱＥＩＢ４１２は、目標スレッドのアドレス５０２と、静止事象のタイプ５０４（すなわち、停止か終了）と、割り込みプログラム状態語（ＰＳＷ）およびレジスタ内容５０６とを記憶するためのアドレス・ロケーションを含むユーザ・アドレス空間において定義された領域である。事象生成ルーチン４０８は、初期的には、ＱＥＩＢ４１２に、目標スレッド・アドレス５０２と静止事象タイプ５０４を格納する。

【００３３】次に、上記サーチによってＴＣＱ３２０で見い出されたスレッド２０４の各々に関し、事象生成ルーチン４０８は、サービス要求ブロック（ＳＲＢ）４１６をスケジュールして（ステップ４１４）、スレッドへ停止要求割込みを送信し、該スレッドを停止させる。ＳＲＢ４１６の各々は、ユーザ・アドレス空間２０６において実行するためカーネルによってディスパッチされる１つの作業単位(unitof work)である。各ＳＲＢ４１６は、目標スレッド２０４に対する割込み要求ブロック（ＩＲＢ）４１８を作成しスケジュールする。ＩＲＢ４１８は、下記のように動作する。

【００３４】すべての該当するスレッド２０４に対する割り込みをスケジュールした後、事象生成ルーチン４０８は、quiesce_freezeサービス４０４に制御を戻す（ステップ４２０）。Quiesce_freezeサービス４０４は、スレッド２０２に関するＴＣＱＥ３０８に配置された事象制御ブロック６０６（図６）上で待機する（ステップ４２２）。この待機は、静止事象に応答する最後のスレッド２０４からのポスト信号によって、最後のスレッドが停止状態に入る直前に解かれる。

【００３５】図７を参照して、上記割り込みメカニズムをより詳細に説明する。ＩＲＢ４１８がアプリケーション２００の制御を獲得すると、実行は、目標スレッド２０４上で（図の符号７０２の箇所で）停止する。ＩＲＢ４１８は、アプリケーションのスレッド状態をチェックして、システム環境が静止事象を扱うことを受容できることを確認する。システム環境が受容可能である場合（ステップ７０４）、ＩＲＢ４１８は、対応するＱＥＩＢ４１２（図５）の部分５０６に、スレッド２０４に関するプログラム状態語（ＰＳＷ）とレジスタ内容を保存する。プログラム状態語（ＰＳＷ）とレジスタ内容は、スレッド２０４に関するシステム・ディスパッチャによって管理される対応タスク・レベル制御ブロック７０８から抽出される。ＰＳＷは、静止事象が処理された後再開されるべき割込み箇所７０２をポイントする。次に、ＩＲＢ４１８は、目標とされたスレッド２０４の再開ＰＳＷを以前に登録した静止出口７１２をポイントするように修正し（ステップ７１０）、レジスタ１をスレッド２０４のＱＥＩＢ４１２をポイントするように修正する（ステップ７１４）。次にプロセスはＩＲＢ４１８から出て（ステップ７１６）、修正されたＰＳＷとレジスタ１を用いて目標スレッド２０４で実行が再開される。

【００３６】静止出口７１２は制御権を獲得しＱＥＩＢ

４１２へのアクセスを有する。静止出口７１２はアプリケーション環境（すなわち、スレッドの実行状態）を検査し、プロセスを停頓させるような重大な資源をスレッド２０４が保持していないことを確認する。

【００３７】システム環境が受容可能であれば（ステップ７１８）、静止出口７１２は、ＱＥＩＢ４１２の静止事象タイプ・フィールド５０４（図５）を調べ、どの事象タイプを処理するため呼び出されたか、また、それ自身を中断または終了させるべきかを判断する。あり得る２つの事象タイプは、スレッド２０４の中断を要求する事象とスレッドの終了を要求する事象とである。本例におけるように、事象がスレッド２０４の中断を要求するタイプである場合、静止出口７１２は、静止サービス２０８のquiesce_freeze_selfサービス７２２を起動する（ステップ７２０）。Quiesce_freeze_selfサービス７２２は、呼び出し元スレッド２０４がアプリケーション２００中で静止状態になるべき最後のスレッドであるかを調べる。そうならば（ステップ７２４）、Quiesce_freeze_selfサービス７２２は、quiesce_freezeサービス４０４に、従って、quiesce_freezeサービスを当初起動したスレッド２０２（図４）に、ポスト信号を送る（ステップ７２６）。Quiesce_freeze_selfサービス７２２は、スレッド２０４を待機状態に入らせるためシステム待機サービスを呼び出すことによってスレッド２０４の実行を中断する（ステップ７２８）。

【００３８】静止出口７１２は、アプリケーション環境が受容できないと判断する場合（ステップ７３０）、quiesce_event_put_backサービス７３４を起動して事象をカーネルへ戻す（ステップ７３２）。Quiesce_event_put_backサービス７３４は、静止事象をこの時点で取り扱うことができないこと、および、アプリケーション２００が後刻事象の送達を要求するであろうこと、をカーネルに通知する。これは、静止事象がなお未処理状態にあることを示すため該当するＴＣＱＥ３０８の事象制御ブロック６０６（図６）にマークをつけることによって行われる。環境から障害を取り除く責任は、アプリケーション２００にある。アプリケーション・スレッド２０４が静止される準備状態になると、Quiesce_freeze_selfサービス７２２を呼び出すか、又は、代替的方法として、静止出口７１２を再起動するようカーネルに要求する。

【００３９】上述のように、すべてのスレッド２０４がｑｕｉｅｓｃｅ－ｆｒｅｅｚｅ－ｓｅｌｆサービス７２２を起動したとき、quiesce_ｆｒｅｅｚｅ－ｓｅｌｆサービス７２６は、静止停止サービス４０４（図４）にポスト信号を送り、制御は、その呼び手（アプリケーション２００のスレッド２０２）に戻される。

【００４０】上述のように、すべてのスレッド２０４がquiesce_freeze_selfサービス７２２を起動した時、quiesce_freeze_selfサービス７２２はquiesce_freezeサービス４０４（図４）にポスト信号を送り、制御は、その呼び手（アプリケーション２００のスレッド２０２）に戻される。図８を参照すると、今やすべての他のスレッド２０４が停止されているので、１つの実行中のスレッド２０２が、アプリケーション２００中の他のスレッド２０４からの干渉なしに、要求される重大な作業を遂行することができる（ステップ８０２）。重大な作業が完了次第、スレッド２０２は、静止サービス２０８のquiesce_unfreezeサービス８０６を呼び出す（ステップ８０４）。quiesce_unfreezeサービス８０６は、pthread_create（図３）の間に構築されたスレッド制御待ち行列エレメント３０８のＴＣＱ３２０を通して動き、quiesce_freeze_selfサービス７２２内で現在待機中のスレッド２０４の各々にポスト信号を送る（ステップ８０８）。ポスト信号発信次第、quiesce_freeze_selfサービスは、各停止スレッド２０４の制御を対応する静止出口７１２に戻し（ステップ８１０）、割込み箇所７０２での実行を再開する（ステップ８１２）。この再開は、ＱＥＩＢ４１２に保存したＰＳＷとレジスタ内容５０６を使用して行われる。次に、quiesce_unfreezeサービス８０６は、スレッド２０２に制御を復帰し（ステップ８１４）、アプリケーション処理を再開する（ステップ８１６）。

【００４１】図９は、静止終了要求のプロセスの流れを示す。静止終了要求は、呼び出し元スレッド２０２を除き、アプリケーション中のスレッド２０４のすべてを終了させる機能を持つ。終了するスレッド２０４は、終了する前にアプリケーション上重大なコードを完了したり、あるいは、スレッド関連資源を問題の起きない状態にする機会を与えられる。

【００４２】図９に示されるように、アプリケーション２００中の他のスレッド２０４を終了させることを要求する事象２１２が通知され次第、スレッド２０２は、アプリケーションの他のスレッドの終了を要求するため、静止サービス９０４のquiesce_termサービス９０４を呼び出す（ステップ９０２）。quiesce_termサービス９０４は、ＴＣＱチェイン３２０を走査し、呼び出し元スレッド２０２に関するＴＣＱＥを除くすべてのチェイン上のＴＣＱＥ３０８の各々に関するquiesce_term事象９０８を生成する。図９には示されていないが、このquiesce_term事象９０８は、第４図で示された方法と同様にサービス要求ブロック（ＳＲＢＳ）と割込み要求ブロック（ＩＲＢＳ）とを使用してスレッド２０４へ伝達されることが好ましい。次に、quiesce_termサービス９０４は、他のスレッド２０４が終了させられるまで、待機する（ステップ９１０）。

【００４３】quiesce_termサービス９０４によって生成された事象９０８をスレッド２０４が受信すると、アプリケーションの通常の流れは、上述の通り箇所７０２で割り込まれ、静止出口７１２に制御権が与えられる。静

止出口７１２は、停止に関し上述したケース（図７）と同様にアプリケーション環境をチェックする。アプリケーション環境が受容可能で、生成された静止事象が（ＱＥＩＢ４１２の静止事象タイプ・フィールド５０４によって標示されるが）終了のためのものであれば（ステップ９１２）、静止出口７１２は、スレッド・クリーンアップ・ルーチン（pthread_exit）９２２を起動する（ステップ９１４）。

【００４４】Pthread_exitルーチン９２２は、終了するスレッド２０４に関係したシステム資源を解放する。当該終了スレッド２０４が、アプリケーション中で、生成されたquiesce_term事象を持った最後のスレッドであれば（ステップ９１６）、pthread_exitルーチン９２２は、quiesce_termサービス９０４で待機中のスレッド２０２にポスト信号を送る（ステップ９１８）。スレッド２０２が上記ポスト信号を受信しquiesce_termサービス９０４での待機を解いた後（ステップ９１８）、アプリケーション２００に制御が戻され、受信した事象２１２に基づいて適切な措置が取られる(ステップ９２０)。

【００４５】（図示されていないが）quiesce_forceサービスが、quiesce_term サービス９０４と同様に動作し、すべての他のスレッド２０４に終了事象を送信する。しかし、quiesce_forceサービスは、ユーザ定義静止出口の呼び出しをバイパスし、割込み要求ブロック（ＩＲＢ）からpthread_exitルーチン９２２が直接呼び出される。

【００４６】まとめとして、本発明の構成に関して以下の事項を開示する。

（１）第１のスレッドと第２のスレッドが共通アドレス空間で並列的に実行されるコンピュータ・システムにおいて、上記第１のスレッドによって外部事象を処理するための方法であって、上記第２のスレッドを静止させるため、上記第１のスレッドから上記第２のスレッドへ静止事象を送信するステップと、上記第２のスレッドへ送信された静止事象に応答して上記第２のスレッドが動作を静止するまで、上記第１のスレッドの実行を中断するステップと、上記第２のスレッドへ送信された静止事象に応答して上記第２のスレッドが動作を静止した時、上記事象を処理するため上記第１のスレッドの実行を再開するステップと、を含む方法。

（２）上記第２のスレッドは、上記アドレス空間において上記第１のスレッドと並列に実行される複数の付加スレッドの１つであり、上記第１のスレッドから上記複数の付加スレッドの各々に上記静止事象が送信される、上記（１）記載の方法。

（３）上記複数の付加スレッドが当該スレッドへ送信された静止事象に応答して動作を静止するまで、上記第１のスレッドの実行が中断される、上記（２）記載の方法。

（４）静止すべき上記複数の付加スレッドの最後のスレッドが、上記第１のスレッドの実行を再開する、上記（３）記載の方法。

（５）上記第１のスレッドによる上記外部事象の検出に応答して、上記静止事象が上記第１のスレッドから上記第２のスレッドへ送信される、上記（１）記載の方法。

（６）上記静止事象が、上記第２のスレッドを終了させるための終了事象である、上記（１）記載の方法。

（７）上記静止事象が、上記第２のスレッドを中断させるための中断事象である、上記（１）記載の方法。

（８）静止事象を上記第１のスレッドから上記第２のスレッドへ送信する上記ステップが、上記第２のスレッドの実行に割り込み、静止出口ルーチンに制御を与えるステップを含む、上記（１）記載の方法。

（９）上記静止出口ルーチンが、上記第２のスレッドがシステム環境に影響を及ぼす程重大な資源を保持しているか否かを判断し、上記第２のスレッドが重大な資源を保持していないと判断する場合、上記第２のスレッドを静止させる、上記（８）記載の方法。

（１０）上記第２のスレッドがシステム環境に影響を及ぼす程重大な資源を保持しているか否かを判断するステップと、上記第２のスレッドが重大な資源を保持していないと判断する場合、上記第２のスレッドを静止させるステップと、を更に含む上記（１）記載の方法。

（１１）上記第２のスレッドを静止させる前に上記第２のスレッドによって保持される重大な資源を解放するステップを更に含む、上記（１０）記載の方法。

（１２）第１のスレッドと第２のスレッドが共通アドレス空間で並列的に実行されるコンピュータ・システムにおいて、上記第１のスレッドによって外部事象を処理するための方法であって、上記第２のスレッドを中断させるため、上記第１のスレッドから上記第２のスレッドへ中断事象を送信するステップと、上記第２のスレッドへ送信された中断事象に応答して上記第２のスレッドが動作を中断するまで、上記第１のスレッドの実行を中断するステップと、上記第２のスレッドへ送信された中断事象に応答して上記第２のスレッドが動作を中断した時、上記事象を処理するため上記第１のスレッドの実行を再開するステップと、上記第１のスレッドによる上記事象の処理の後、上記第２のスレッドの実行を再開するステップと、を含む方法。

（１３）複数の付加スレッドが上記アドレス空間において上記第１のスレッドと並列に実行され、上記第１のスレッドから上記複数の付加スレッドの各々に上記中断事象が送信される、上記（１２）記載の方法。

（１４）上記複数の付加スレッドが当該スレッドへ送信された中断事象に応答して動作を中断するまで、上記第１のスレッドの実行が中断される、上記（１３）記載の方法。

【００４７】

【発明の効果】本発明によって、静止状態がいつスレッ

ドに効果を及ぼす状態に入るべきかを決定することが可能となる。また、重大な資源を保持しているスレッドを無条件に停止させるという問題が回避される。コードの「不安定」または「重大な」部分で実行中のスレッドを非同期的に停止させることに起因する破壊的な結果もまた回避される。

【図面の簡単な説明】

【図１】本発明を組み入れているコンピュータ・システムの概要を示す図である。

【図２】本発明に従って他のスレッドを静止（すなわち停止または終了）させるための一般的手続きを示す図である。

【図３】本発明に従って静止出口を初期登録するための手続きを示す図である。

【図４】本発明に従って他のスレッドを停止（すなわち、中断 ）する手続きの初期局面を示す図である。

【図５】図４で示される停止手続きの初期局面の間に作成される静止出口インターフェース・ブロック（ＱＥＩＢ）を示す図である。

【図６】図３で示される初期登録手続きの間に作成されるスレッド制御待ち行列エレメント（ＴＣＱＥ）を示す図である。

【図７】本発明に従って他のスレッドを停止させる手続きの最終局面を示す図である。

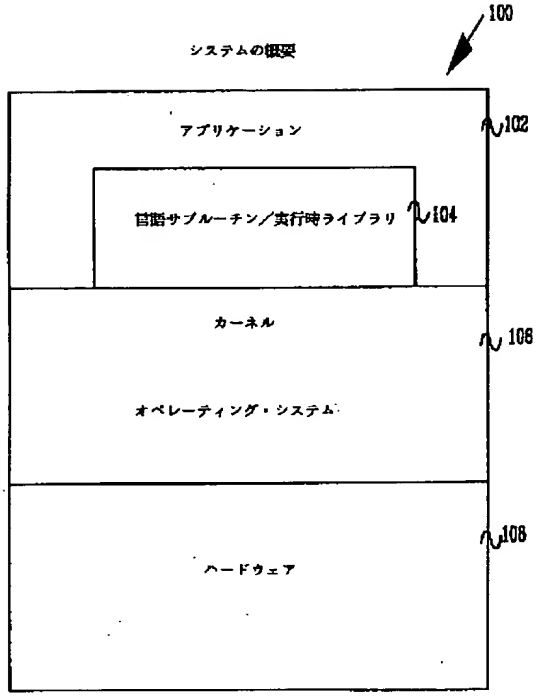
【図８】本発明に従って他のスレッドの停止を解く（すなわち、再開する）ための手続きを示す図である。

【図９】本発明に従って他のスレッドを終了させるための手続きを示す図である。

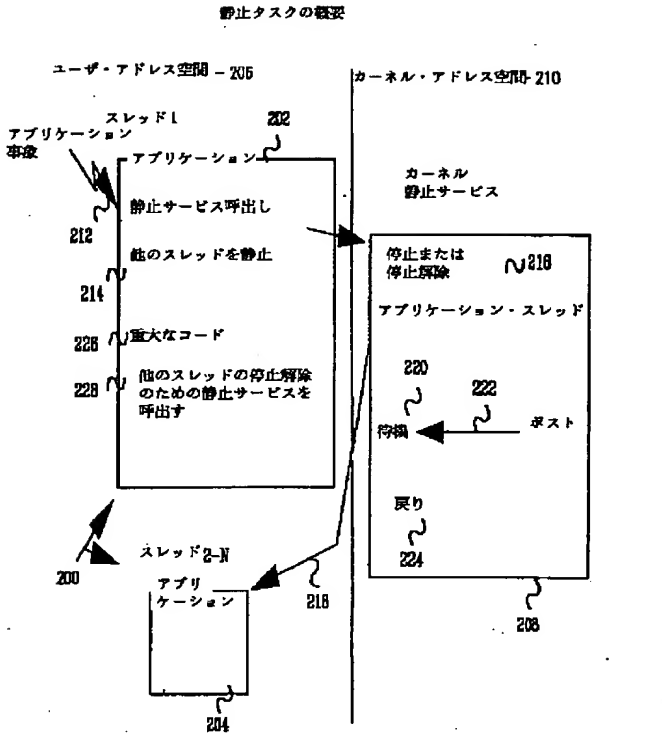
【符号の説明】

１００	コンピュータ・システム
１０２	アプリケーション層
１０４	言語サブルーチン／実行ライブラリ層
１０６	オペレーティング・システムまたはカーネル層
１０８	ハードウエア層
２００	アプリケーション
２０２、３０２	第１のスレッド
２０４、３０４、３１２	第２のスレッド
２０６	ユーザ・アドレス空間
２０８	静止サービス
２１２	アプリケーション事象
２１８	静止通知
３０６	quiesce_exit_registration（出口登録）サービス
３０８	スレッド制御待ち行列エレメント（ＴＣＱＥ）
３１４	pthread_create（スレッド作成）サービス
３２０	スレッド制御待ち行列（ＴＣＱ）
４０４	quiesce_freeze（停止）サービス
４０８	内部事象生成ルーチン
４１２	静止出口インターフェース・ブロック（ＱＥＩＢ）
４１６	サービス要求ブロック（ＳＲＢ）
４１８	割り込み要求ブロック（ＩＲＢ）
５０２	目標スレッド・アドレス
５０４	静止事象タイプ
５０６	ＰＳＷとレジスタ内容
６０２	スレッド識別子
６０４	アプリケーション静止出口アドレス
６０６	事象制御ブロック
７０２	割り込み箇所
７１２	静止出口
７２２	quiesce_freeze_self（自己停止）サービス
７３４	quiesce_event_put_back（処理遅延）サービス
８０６	quiesce_unfreeze（停止解除）サービス
９０４	quiesce_term（終了）サービス
９０８	quiesce_term事象
９２２	スレッド・クリーンアップ・ルーチン（pthread_exit）

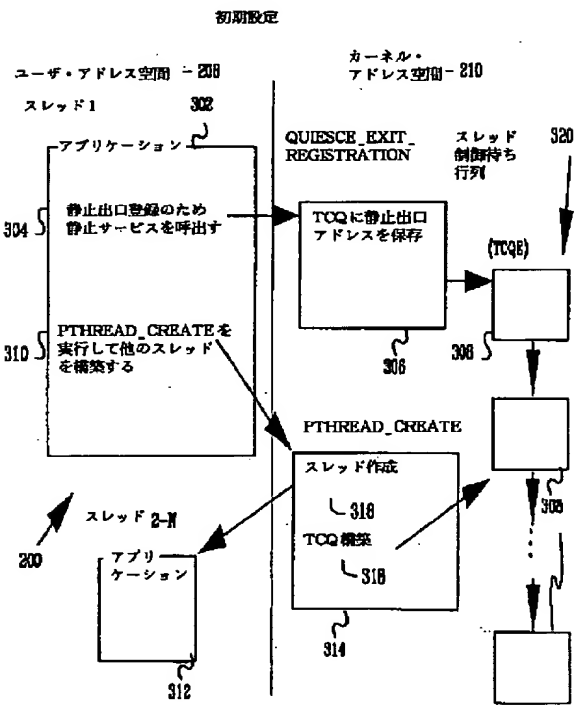
【図１】


100
システムの概要
アプリケーション
102
言語サブルーチン/実行時ライブラリ
104
カーネル
106
オペレーティング・システム
ハードウェア
108

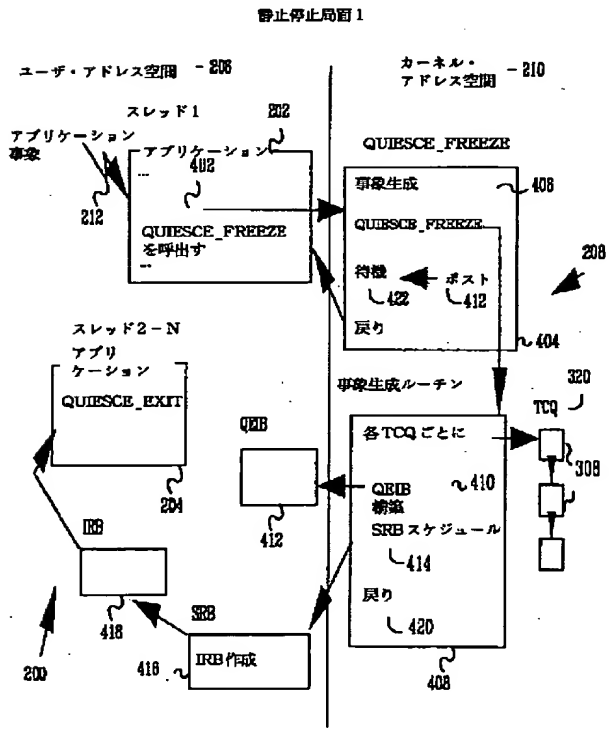
【図２】


静止タスクの概要
ユーザ・アドレス空間 - 205
カーネル・アドレス空間- 210
スレッド1
202
アプリケーション事象
アプリケーション
静止サービス呼出し
212
他のスレッドを静止
214
重大なコード
226
他のスレッドの停止解除のための静止サービスを呼出す
228
カーネル静止サービス
停止または停止解除
216
アプリケーション・スレッド
220
222
待機
ポスト
戻り
224
208
200
スレッド2-N
アプリケーション
218
204

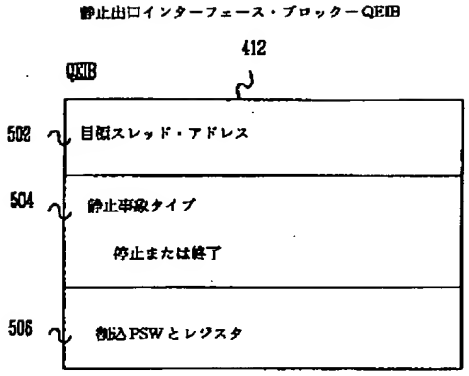
【図３】


初期設定
ユーザ・アドレス空間 - 205
カーネル・アドレス空間- 210
スレッド1
302
アプリケーション
QUIESCE_EXIT_REGISTRATION
スレッド制御待ち行列
320
304
静止出口登録のため静止サービスを呼出す
TCQに静止出口アドレスを保存
(TCQE)
306
308
310
PTHREAD_CREATEを実行して他のスレッドを構築する
PTHREAD_CREATE
スレッド作成
318
TCQ構築
316
305
スレッド2-N
200
アプリケーション
312
314

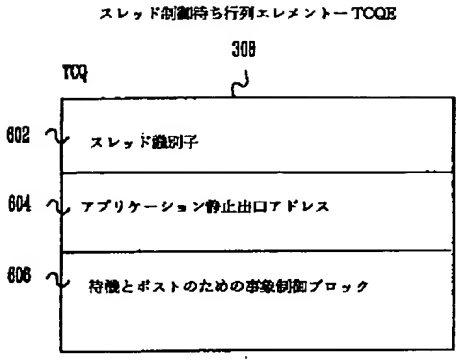
【図４】


静止停止局面1
ユーザ・アドレス空間 - 205
カーネル・アドレス空間 - 210
スレッド1
202
アプリケーション事象
アプリケーション
402
212
QUIESCE_FREEZEを呼出す
QUIESCE_FREEZE
事象生成
406
QUIESCE_FREEZE
待機
422
ポスト
412
戻り
404
208
スレッド2-N
アプリケーション
QUIESCE_EXIT
204
事象生成ルーチン
320
TCQ
各TCQごとに
308
QEIB
412
QEIB構築
410
SRBスケジュール
414
戻り
420
408
IRB
418
SRB
IRB作成
416
200

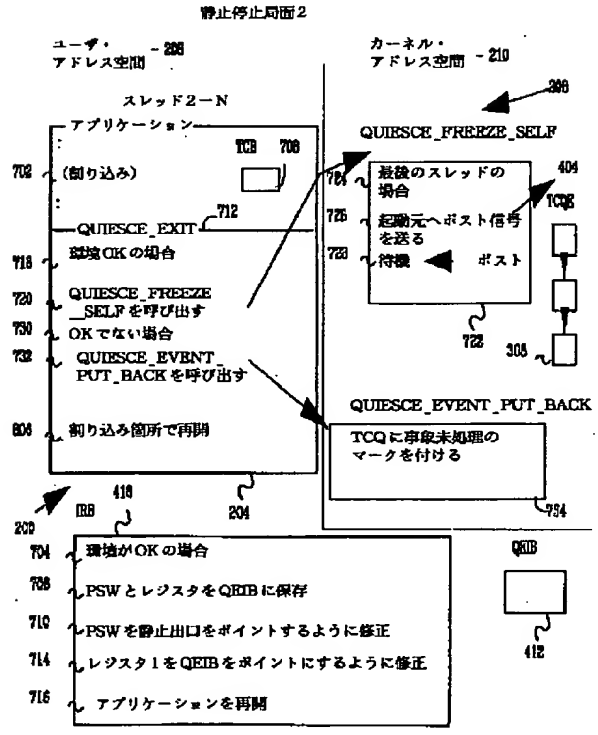
【図５】


静止出口インターフェース・ブロックーQEIB
412
QEIB
502
目標スレッド・アドレス
504
静止事象タイプ
停止または終了
506
割込PSWとレジスタ

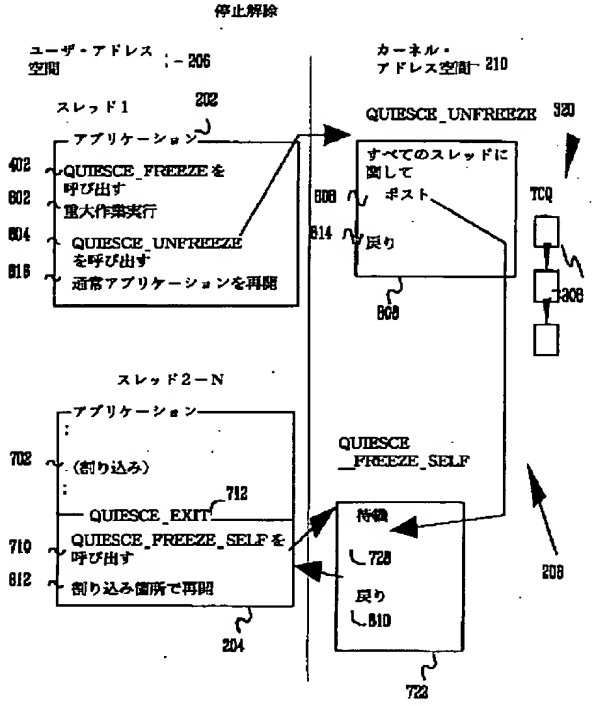
【図６】


スレッド制御待ち行列エレメントーTCQE
309
TCQ
602
スレッド識別子
604
アプリケーション静止出口アドレス
606
待機とポストのための事象制御ブロック

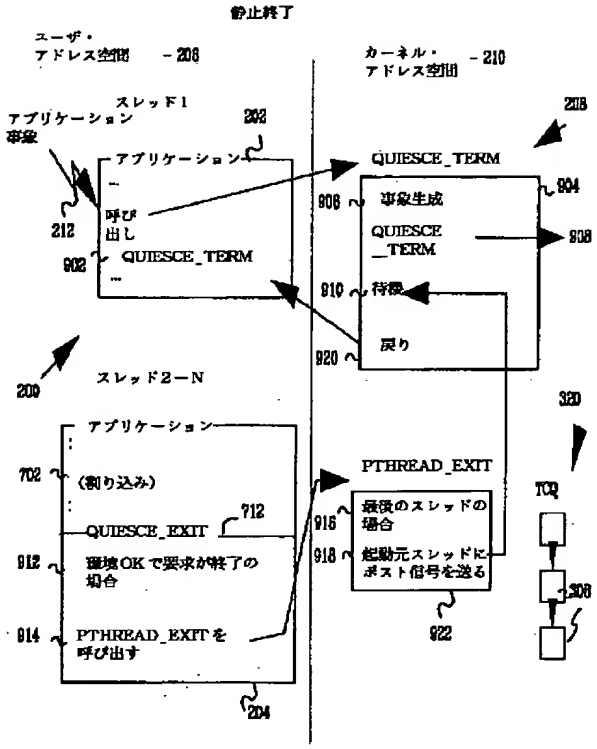
【図７】


静止停止局面2
ユーザ・アドレス空間 206
カーネル・アドレス空間 210
スレッド2ーN
アプリケーション
TCB 708
702 (割り込み)
712
QUIESCE_EXIT
718 環境OKの場合
720 QUIESCE_FREEZE_SELFを呼び出す
730 OKでない場合
732 QUIESCE_EVENT_PUT_BACKを呼び出す
804 割り込み箇所で再開
208
IRB 416
204
704 環境がOKの場合
706 PSWとレジスタをQEIBに保存
710 PSWを静止出口をポイントするように修正
714 レジスタ1をQEIBをポイントにするように修正
716 アプリケーションを再開
QEIB
412
306
QUIESCE_FREEZE_SELF
724 最後のスレッドの場合
726 起動元へポスト信号を送る
728 待機 ポスト
404
TCQE
722
305
QUIESCE_EVENT_PUT_BACK
TCQに事象未処理のマークを付ける
734

【図８】


停止解除
ユーザ・アドレス空間 206
カーネル・アドレス空間 210
スレッド1
202
アプリケーション
402 QUIESCE_FREEZEを呼び出す
602 重大作業実行
604 QUIESCE_UNFREEZEを呼び出す
816 通常アプリケーションを再開
QUIESCE_UNFREEZE 320
すべてのスレッドに関して
606 ポスト
614 戻り
608
TCQ
308
スレッド2ーN
アプリケーション
702 (割り込み)
712
QUIESCE_EXIT
710 QUIESCE_FREEZE_SELFを呼び出す
812 割り込み箇所で再開
204
QUIESCE_FREEZE_SELF
待機
728
戻り
610
208
722

【図９】



フロントページの続き

(72)発明者　アーネスト・スコット・ベンダー
アメリカ合衆国12477ニューヨーク州ソーガティース、パイン・グローブ・スクール・ロード　27

(72)発明者　ジョン・ケビン・フランクス
アメリカ合衆国12477ニューヨーク州ソーガティース、デーブ・エリオット・ロード　186

(72)発明者　ジョン・アーサー・ヘルムボルド
アメリカ合衆国12401ニューヨーク州キングストン、リンダーマン・アベニュー　583